FleetOutlook® Data Pump User Guide

Updated 12/4/2013

CalAmp Corporation
13645 Dulles Technology Drive, Herndon, VA 20171
Phone : (703) 262-0500
FAX : (703) 262-0380
www.calamp.com

Document: MBUD-0189v3.1
December 2013

FleetOutlook | Data Pump User Guide

Revision History:		
Version #	Revision Date	Details
0.1	10/20/2008	First draft, revised from older document.
0.2	10/21/2008	Sample web service call details
1.0	12/08/2008	Added Replay Messages operation details
2.0	10/05/2010	Updated with usage clarifications
3.0	10/2013	Updated for OBD and JBUS data
3.1	12/04/2013	Updated with Alert data

TABLE OF CONTENTS

1 PURPOSE

This document explains the Data Pump Service. This service enables a customer to receive live event data for vehicles that have active devices reporting to FleetOutlook.

1.1 DOCUMENT SCOPE

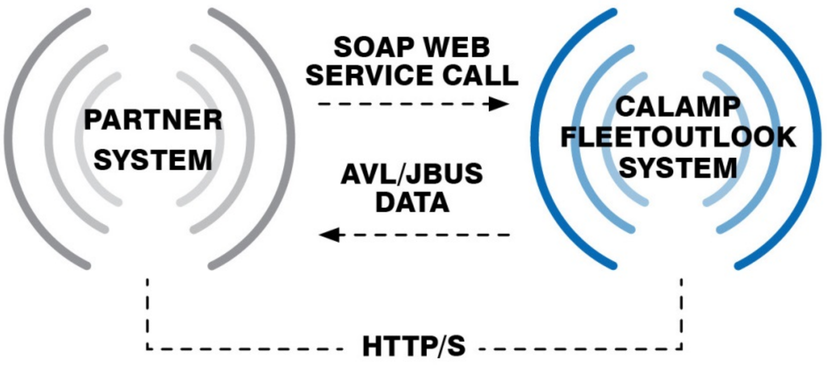
The document provides an overview of the Data Pump Service. The audience includes partners who need to interface and interact with the Data Pump Service.

2 DATA PUMP OVERVIEW

The Data Pump service is a standard SOAP messaging protocol in a web service executed over HTTP/S. The Data Pump service enables a read-only call to pull operational data from FleetOutlook directly to your business for use with third-party systems. The format of the response is an XML schema structure containing the most recent event messages.

Using the Data Pump service ensures that your entire enterprise receives intelligence about your fleet operations in a format that allows maximum use with the systems that manage your business.

Sample Data Returned	
AVL Data	**OBD-II, J1708 and J1939 (JBUS) Data**
Event Time	Fuel Usage
GPS Location (Latitude and Longitude)	Engine Hours
Event Address	Oil Temperature
Speed	Coolant Temperature



2.1 DATA PUMP USAGE

CalAmp provides the Data Pump service to FleetOutlook customers who want an automated process to obtain live event message data from vehicles reporting periodically. This process enables customers to utilize the data within their own applications and reporting systems, in addition to employing the FleetOutlook application.

Each vehicle reporting to FleetOutlook sends an event message containing AVL and/or JBUS data such as speed, heading, fuel usage and input state. FleetOutlook processes these event messages and then plots the vehicle location history on tracking maps as well as generates alerts based on predefined conditions. The Data Pump service enables customers to retrieve the data from the raw event messages in a standard XML format.

The Customer System requests the data using a SOAP messaging protocol in a web service call. The SOAP messaging protocol enables communication between two systems using HTTP/S and XML. FleetOutlook temporarily stores the event messages for an entire enterprise in an enterprise specific queue until the Customer System enables a SOAP web service call (i.e., getEvents). FleetOutlook returns the event messages stored in the queue. Once pulled, an event message is no longer available in the queue.

2.2 AUTHENTICATION

The FleetOutlook SOAP web service requires WS-Security Username Token format where password type is PasswordText. The web service requires authentication in each call from the Customer System. Login credentials are sent in the SOAP header.

Note: CalAmp provides the login credentials to each Customer System. The login credentials are for the SOAP web service not for the FleetOutlook application.

2.3 DATA PUMP QUEUES

Each Customer System may have one or more queues depending on an enterprise's organizational needs. Each queue requires authentication at the time of a call using the WS-Security UsernameToken format, and each web service call is handled as an independent transaction. The polling frequency set by the Customer System is dependent on the enterprise size and number of active devices. CalAmp does not recommend invoking calls more than once per minute.

The Customer System defines the number of event messages returned for each call. If the parameter is not defined, the default number is 100 event messages.

2.4 CAPACITY

Currently, each queue is provisioned to hold approximately 25,000 event messages. All Customer System queues are stored on shared infrastructure and queue size is monitored for stability.

2.5 MONITORING

The CalAmp Network Operations Center (NOC) actively monitors the queues for all our Customer Systems. NOC alerts a Customer System when the number of event messages in a queue exceeds the threshold. Each Customer System has a grace period of four hours to begin invoking web service calls to retrieve event messages. In the event the grace period has expired and the Customer System's queue is still above threshold, CalAmp may delete undelivered event messages in a queue.

Note: The current monitoring system only allows CalAmp to remove all event messages stored in a queue. CalAmp cannot remove a subset of event messages.

Refer to **CalAmp Capacity Policies** for detailed information on event message storage and queue threshold.

2.5.1 CALAMP CAPACITY POLICIES

The Data Pump message queue is not a long-term storage container for an enterprise's event messages. The CalAmp NOC (Network Operations Center) monitors each queue. To ensure optimal performance, CalAmp adheres to the following procedures and policies.

- Event messages stored in a Customer System's queue are temporary. CalAmp expects each Customer System to pull event messages on a regular basis. Pulling frequency is dependent on enterprise size and number of active devices.
- CalAmp requests prior notification for any outage on the Customer System. Based on expected outage duration, CalAmp will adjust the queue threshold assigned to the Customer System.
- CalAmp's NOC (Network Operations Center) actively monitors all queues. In the event that a Customer System queue is above the threshold, NOC notifies the customer to pull event messages.
- In the event the Customer System is unresponsive to NOC alerts and queue continues to exceed, CalAmp may delete undelivered event messages stored in the queue and stop delivery of new messages to the queue.

3 WEB SERVICE CALLS

CalAmp supports two web service calls: getEvents and getMessageCount. getEvents enables the Customer System to pull a user-specified number of event messages from a queue. getMessageCount enables the Customer System to monitor the number of event messages currently in their queue. Additionally, getMessageCount enables the Customer System to ensure their queue does not grow to an excessive number.

WSDL Operation: getEvents

The getEvents WSDL operation does not require any additional input parameters; however, if you do not specify a number of event messages to pull, the default return number is 100.

Field	Description	Format	Optional
Messages	Number of messages to pull	Integer	Yes

WSDL Operation: getMessageCount

The getMessageCount WSDL operation does not require any additional input parameters.

4 FLEETOUTLOOK OUTPUT

FleetOutlook returns data in the web service call, getEventsResponse. The getEventReponse web service returns all AVLEvents, JBUSEvents or AlertEvents, depending on the device capability and customer specific configurations.

WSDL Operation: getEventsResponse

AVLEvent

AVLEvent		
Field	**Description**	**Format**
deviceID		String
vehicleID		String
driverID		String
vehicleVIN		String
driverID		String
event		Event
GMTTime		DateTime
timeOffset	Offset from GMTTime to the enterprise's FleetOutlook configured time zone.	Integer
eventype	AVL events such as moving, stopped, etc.	String
GPS		GPS
GPSVAlidity	True or False. Indicates satellite fix.	Boolean
latitude		Double
longtitude		Double
address		address
street		String
crossStreet		String
city		String
state		String
zip		String
country		String
telemetry		telemetry
vehicleSpeed		Integer
heading	Moving direction.	Integer
odometer		Double
engineHours	Reported for vehicles using an OBD-II device.	Double
fuelUsage	Reported for vehicles using an OBD-II device.	Double
digitalIO		digitalIO
messageSeqID	Unique ID by enterprise, which enables you to detect missing messages.	Long
messageDataID	Unique ID across the entire system. Used to troubleshoot message issues.	Long

WSDL Operation: getEventsResponse

JBUSEvent

JBUSEvent		
Field	**Description**	**Format**
GMTTime		DateTime
deviceID		String
vehicleID		String
vehicleVIN		String
driverID		String
GPS		GPS
JBUSMessage		JBUSMessage
jbusProtocol	Indicates either 1708 or 1939.	String
vin		String
Odometer	Vehicle's current odometer reading	Double
High Resolution Odometer	Only in J1939 protocol.	Double
Battery Voltage		Double
Switched Battery Voltage		Double
Engine Speed	RPM.	Integer
Total Fuel	Lifetime reading of fuel consumed by the engine.	Double
Total Idle Fuel	Lifetime reading of fuel consumed by the engine while not in motion.	Double
Total Idle Hours	Lifetime number of hours spent with the engine on while not in motion.	Integer
Total Engine Hours	Lifetime number of hours has been on.	Integer
Engine Coolant Temp		Integer
Engine Oil Temp		Double
Seat Belt Used	Only in J1939 protocol.	Boolean

WSDL Operation: getEventsResponse

AlertEvent

AlertEvent		
Field	**Description**	**Format**
transactionID		String
alertName		String
alertType*		String
GMTTime		DateTime
group		String
deviceID		String
vehicleID		String
driverID		String
address		String
landmarkName		String
landmarkCategory		String
stopTimeThreshold		Integer

*Note: Currently, data feed supports the following 3 alert types: Entering Landmark, Stopped at Landmark and Leaving Landmark.

4.1 MESSAGE SEQUENCE

Each message has a unique Message ID (i.e., XML tag is <MessageDataID>). The Customer Systems use the MessageDataID field to detect and remove duplicate event messages.

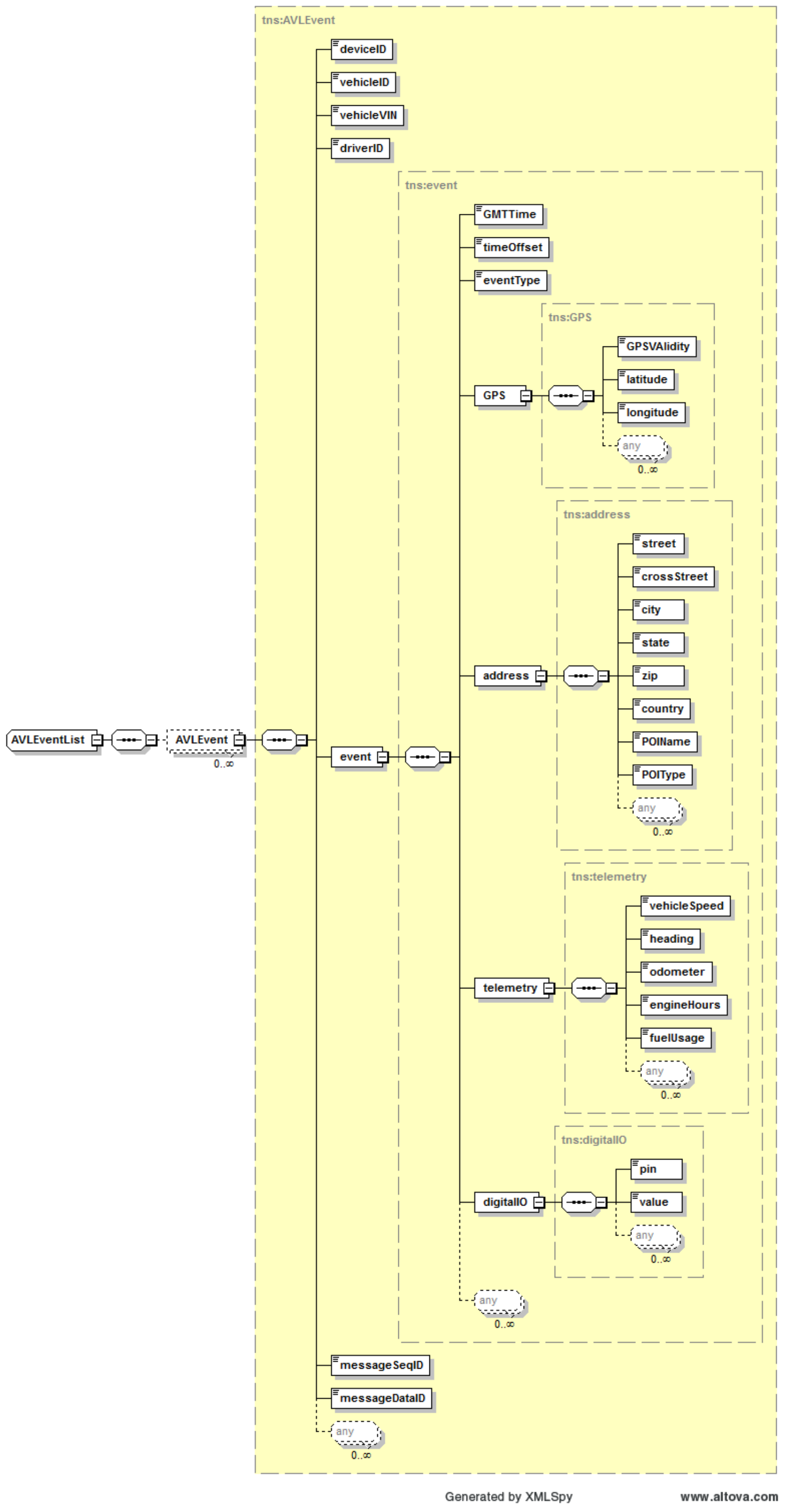
Each message also carries a per-company, non-decreasing consecutive number (Sequence Number) in a tag (i.e., XML tag is <MsgSeqID>). Using the sequence number, the Customer System can determine if any event message is missing. For example, MsgSeqID 708 is missing if the Customer System received MsgSeqID 707 and MsgSeqID 709.

5 APPENDIX: FLEETOUTLOOK SAMPLE XML OUTPUT

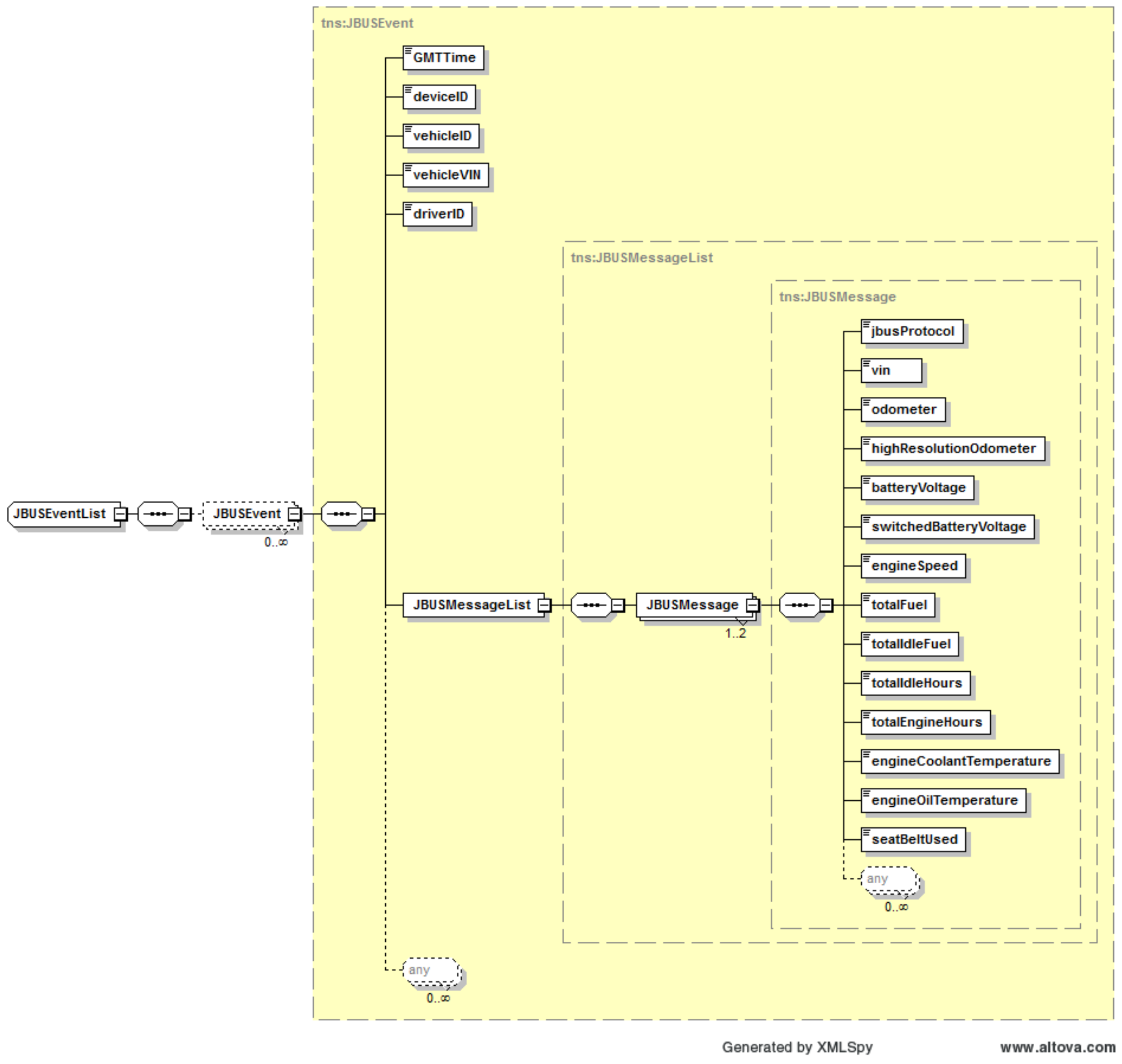
```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:getMessagesResponse xmlns:ns2="http://calamp.com/DataFeedService/">
      <AVLEvents>1</AVLEvents>
      <JBUSEvents>0</JBUSEvents>
      <AVLEventList>
        <AVLEvent>
          <deviceID>4160003764</deviceID>
          <vehicleID>3764-QA</vehicleID>
          <vehicleVIN xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"/>
          <driverID>93560</driverID>
          <event>
            <GMTTime>2013-10-02T22:01:24.000Z</GMTTime>
            <timeOffset>-4</timeOffset>
            <eventType>MOVING</eventType>
            <GPS>
              <GPSVAlidity>true</GPSVAlidity>
              <latitude>37.52278</latitude>
              <longitude>-122.0005</longitude>
            </GPS>
            <address>
              <street>39446 Cedar Blvd</street>
              <crossStreet xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"/>
              <city>Newark</city>
              <state>California</state>
              <zip>94560</zip>
              <country>USA</country>
              <POIName>146171</POIName>
              <POIType>n/a</POIType>
            </address>
            <telemetry>
              <vehicleSpeed>0</vehicleSpeed>
              <heading>0</heading>
              <odometer>308781.7</odometer>
              <engineHours>3606.78</engineHours>
              <fuelUsage>0.0</fuelUsage>
            </telemetry>
            <digitalIO xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"/>
```

```
          </event>
          <messageSeqID>333861</messageSeqID>
          <messageDataID>53973640082</messageDataID>
        </AVLEvent>
      </AVLEventList>
      <JBUSEventList xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"/>
      <errorMessage xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"/>
    </ns2:getMessagesResponse>
  </soap:Body>
</soap:Envelope>
```

6 APPENDIX: XML OUTPUT DIAGRAM – AVL EVENTS



7 APPENDIX: XML OUTPUT DIAGRAM – JBUS EVENTS



8 APPENDIX: SAMPLE WEB SERVICE CALL

```
<soapenv:Envelope xmlns:dat="http://calamp.com/DataFeedService/"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
      <wsse:UsernameToken wsu:Id="UsernameToken-1">
        <wsse:Username>Chase</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordText">password123</wsse:Password>
        <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-soap-message-security-
1.0#Base64Binary">+QzImhlYMaV7+olL1KGJaQ==</wsse:Nonce>
        <wsu:Created>2013-09-19T17:34:39.753Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <dat:getMessagesRequest>
      <!--Optional:-->
      <messages>1</messages>
    </dat:getMessagesRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

9 APPENDIX: WSDL SAMPLE

Refer to the following WSDL link for assistance – http://www.wrx-us.net/datafeedservice/1013/DataFeedService?wsdl

```
<?xml version='1.0' encoding='UTF-8'?><wsdl:definitions
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="http://calamp.com/DataFeedService/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="DataFeedService"
targetNamespace="http://calamp.com/DataFeedService/">
  <wsdl:types>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://calamp.com/DataFeedService/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://calamp.com/DataFeedService/">
      <xsd:element name="getMessagesRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element minOccurs="0" name="messages" type="xsd:int"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="getMessagesResponse">
        <xsd:complexType>
          <xsd:sequence>
               <xsd:element name="AlertEvents" nillable="true" type="xsd:int"/>
            <xsd:element name="AVLEvents" nillable="true" type="xsd:int"/>
            <xsd:element name="JBUSEvents" nillable="true" type="xsd:int"/>
            <xsd:element name="AlertEventList" nillable="true" type="tns:AlertEventList"/>
            <xsd:element name="AVLEventList" nillable="true" type="tns:AVLEventList"/>
            <xsd:element name="JBUSEventList" nillable="true" type="tns:JBUSEventList"/>
            <xsd:element name="errorMessage" nillable="true" type="xsd:string"/>
            <xsd:any maxOccurs="unbounded" minOccurs="0" processContents="skip"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:complexType name="AlertEventList">
        <xsd:sequence>
          <xsd:element maxOccurs="unbounded" minOccurs="0" name="AlertEvent"
type="tns:AlertEvent"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="AVLEventList">
        <xsd:sequence>
          <xsd:element maxOccurs="unbounded" minOccurs="0" name="AVLEvent" type="tns:AVLEvent"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="JBUSEventList">
        <xsd:sequence>
          <xsd:element maxOccurs="unbounded" minOccurs="0" name="JBUSEvent"
type="tns:JBUSEvent"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="AlertEvent">
        <xsd:sequence>
          <xsd:element name="transactionId" nillable="true" type="xsd:string"/>
          <xsd:element name="alertName" nillable="true" type="xsd:string"/>
          <xsd:element name="alertType" nillable="true" type="xsd:string"/>
         <xsd:element name="GMTTime" nillable="true" type="xsd:dateTime"/>
         <xsd:element name="group" nillable="true" type="xsd:string"/>
          <xsd:element name="deviceID" nillable="true" type="xsd:string"/>
          <xsd:element name="vehicleID" nillable="true" type="xsd:string"/>
          <xsd:element name="driverID" nillable="true" type="xsd:string"/>
          <xsd:element name="address" nillable="true" type="xsd:string"/>
          <xsd:element name="landmarkName" nillable="true" type="xsd:string"/>
          <xsd:element name="landmarkCategory" nillable="true" type="xsd:string"/>
          <xsd:element name="stopTimeThreshold" nillable="true" type="xsd:int"/>
          <xsd:any maxOccurs="unbounded" minOccurs="0" processContents="skip"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="AVLEvent">
        <xsd:sequence>
          <xsd:element name="deviceID" nillable="true" type="xsd:string"/>
          <xsd:element name="vehicleID" nillable="true" type="xsd:string"/>
          <xsd:element name="vehicleVIN" nillable="true" type="xsd:string"/>
          <xsd:element name="driverID" nillable="true" type="xsd:string"/>
          <xsd:element name="event" nillable="true" type="tns:event"/>
          <xsd:element name="messageSeqID" nillable="true" type="xsd:long"/>
          <xsd:element name="messageDataID" nillable="true" type="xsd:long"/>
          <xsd:any maxOccurs="unbounded" minOccurs="0" processContents="skip"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="event">
        <xsd:sequence>
          <xsd:element name="GMTTime" nillable="true" type="xsd:dateTime"/>
          <xsd:element name="timeOffset" nillable="true" type="xsd:int"/>
          <xsd:element name="eventType" nillable="true" type="xsd:string"/>
          <xsd:element name="GPS" nillable="true" type="tns:GPS"/>
          <xsd:element name="address" nillable="true" type="tns:address"/>
          <xsd:element name="telemetry" nillable="true" type="tns:telemetry"/>
          <xsd:element name="digitalIO" nillable="true" type="tns:digitalIO"/>
          <xsd:any maxOccurs="unbounded" minOccurs="0" processContents="skip"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="GPS">
        <xsd:sequence>
          <xsd:element name="GPSVAlidity" nillable="true" type="xsd:boolean"/>
          <xsd:element name="latitude" nillable="true" type="xsd:double"/>
          <xsd:element name="longitude" nillable="true" type="xsd:double"/>
          <xsd:any maxOccurs="unbounded" minOccurs="0" processContents="skip"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="address">
        <xsd:sequence>
          <xsd:element name="street" nillable="true" type="xsd:string"/>
          <xsd:element name="crossStreet" nillable="true" type="xsd:string"/>
          <xsd:element name="city" nillable="true" type="xsd:string"/>
          <xsd:element name="state" nillable="true" type="xsd:string"/>
          <xsd:element name="zip" nillable="true" type="xsd:string"/>
          <xsd:element name="country" nillable="true" type="xsd:string"/>
          <xsd:element name="POIName" nillable="true" type="xsd:string"/>
          <xsd:element name="POIType" nillable="true" type="xsd:string"/>
          <xsd:any maxOccurs="unbounded" minOccurs="0" processContents="skip"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="telemetry">
        <xsd:sequence>
          <xsd:element name="vehicleSpeed" nillable="true" type="xsd:int"/>
          <xsd:element name="heading" nillable="true" type="xsd:int"/>
          <xsd:element name="odometer" nillable="true" type="xsd:double"/>
          <xsd:element name="engineHours" nillable="true" type="xsd:double"/>
          <xsd:element name="fuelUsage" nillable="true" type="xsd:double"/>
          <xsd:any maxOccurs="unbounded" minOccurs="0" processContents="skip"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="digitalIO">
        <xsd:sequence>
          <xsd:element name="pin" nillable="true" type="xsd:string"/>
          <xsd:element name="value" nillable="true" type="xsd:string"/>
          <xsd:any maxOccurs="unbounded" minOccurs="0" processContents="skip"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="JBUSEvent">
        <xsd:sequence>
          <xsd:element name="GMTTime" nillable="true" type="xsd:dateTime"/>
          <xsd:element name="deviceID" nillable="true" type="xsd:string"/>
          <xsd:element name="vehicleID" nillable="true" type="xsd:string"/>
          <xsd:element name="vehicleVIN" nillable="true" type="xsd:string"/>
          <xsd:element name="driverID" nillable="true" type="xsd:string"/>
```

```
          <xsd:element name="JBUSMessageList" nillable="true" type="tns:JBUSMessageList"/>
          <xsd:any maxOccurs="unbounded" minOccurs="0" processContents="skip"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="JBUSMessageList">
        <xsd:sequence>
          <xsd:element maxOccurs="2" name="JBUSMessage" type="tns:JBUSMessage"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="JBUSMessage">
        <xsd:sequence>
          <xsd:element maxOccurs="1" name="jbusProtocol" nillable="true"
type="tns:JBusProtocol"/>
          <xsd:element name="vin" nillable="true" type="xsd:string"/>
          <xsd:element name="odometer" nillable="true" type="xsd:double"/>
          <xsd:element name="highResolutionOdometer" nillable="true" type="xsd:double"/>
          <xsd:element name="batteryVoltage" nillable="true" type="xsd:double"/>
          <xsd:element name="switchedBatteryVoltage" nillable="true" type="xsd:double"/>
          <xsd:element name="engineSpeed" nillable="true" type="xsd:double"/>
          <xsd:element name="totalFuel" nillable="true" type="xsd:double"/>
          <xsd:element name="totalIdleFuel" nillable="true" type="xsd:double"/>
          <xsd:element name="totalIdleHours" nillable="true" type="xsd:double"/>
          <xsd:element name="totalEngineHours" nillable="true" type="xsd:double"/>
          <xsd:element name="engineCoolantTemperature" nillable="true" type="xsd:int"/>
          <xsd:element name="engineOilTemperature" nillable="true" type="xsd:double"/>
          <xsd:element name="seatBeltUsed" nillable="true" type="xsd:boolean"/>
          <xsd:any maxOccurs="unbounded" minOccurs="0" processContents="skip"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:simpleType name="JBusProtocol">
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="1708"/>
          <xsd:enumeration value="1939"/>
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:element name="getMessageCountRequest">
        <xsd:complexType/>
      </xsd:element>
      <xsd:element name="getMessageCountResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="messageCount" nillable="true" type="xsd:long"/>
            <xsd:element name="errorMessage" nillable="true" type="xsd:string"/>
            <xsd:any maxOccurs="unbounded" minOccurs="0" processContents="skip"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="getMessagesResponse">
    <wsdl:part element="tns:getMessagesResponse" name="getMessagesResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getMessageCountRequest">
    <wsdl:part element="tns:getMessageCountRequest" name="getMessageCountRequest">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getMessagesRequest">
    <wsdl:part element="tns:getMessagesRequest" name="getMessagesRequest">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getMessageCountResponse">
    <wsdl:part element="tns:getMessageCountResponse" name="getMessageCountResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="DataFeedService">
    <wsdl:operation name="getMessages">
      <wsdl:input message="tns:getMessagesRequest">
    </wsdl:input>
      <wsdl:output message="tns:getMessagesResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getMessageCount">
      <wsdl:input message="tns:getMessageCountRequest">
    </wsdl:input>
      <wsdl:output message="tns:getMessageCountResponse">
    </wsdl:output>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="DataFeedServiceServiceSoapBinding" type="tns:DataFeedService">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getMessages">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getMessageCount">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="DataFeedServiceService">
    <wsdl:port binding="tns:DataFeedServiceServiceSoapBinding" name="DataFeedServicePort">
      <soap:address location="http://qa.wrx-us.net/datafeedservice/1013/DataFeedService"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions
```